Vector Software

WHITEPAPER

Mit VectorCAST Software nach ISO 26262 verifizieren und validiere

Aufgabe

Dieses Dokument soll als Referenz dienen um zu zeigen, wie die VectorCAST Produkte von Vector Software eingesetzt werden können, um die Anforderungen an die Verifizierung und Validierung wie sie im ISO 26262 Standard definiert sind einzuhalten, welche selbst von IEC 61508 abgeleitet wurden. Es ist nicht Ziel des Dokumentes, eine umfassende Rezession des Standards wiederzugeben, sondern einen Überblick über die verschiedenen Anforderungen zu verschaffen, welche durch den Einsatz von VectorCAST eingehalten werden können. Für weitere Informationen nehmen Sie bitte mit Vector Software Kontakt auf.

Einführung

Der Wettbewerb in der Automobilindustrie ist sehr hoch. Unternehmen müssen ständig Neuerungen vornehmen um am Markt erfolgreich zu sein indem sie neue Produkteigenschaften einführen, von denen viele signifikante Mengen an Software mitbringen. Das Automobil hat sich heute aus einer primär mechanischen Vorrichtung in ein integriertes Gerät mit eingebetteter Software in allen Hauptsystemen inklusive der Motorsteuerung, dem Antriebsstrang, das Fahrwerk, die Bremsen und nicht zuletzt dem Infotainment, entwickelt.

Die Kosten für eingebettete Systeme in Automobilen zu kontrollieren ist besonders wichtig für Anbieter im Bereich der automobilen Industrie, da es ein viel höheres Volumen von Software gibt als in anderen Industrien mit sicherheitskritischen Anwendungen wie zum Beispiel der Bordelektronik in der Bahn.

Das Testen von Software ist traditionell eine eher teure Angelegenheit, jedoch sind die Kosten für das Auffinden von Software Bugs sehr viel geringer als die Kosten, die durch beschädigte Markenzeichen (infolge von z.B. Rückrufen) entstehen. Das Testen ist deshalb heutzutage eine absolute Notwendigkeit in der Automobilindustrie.

Standards der Software Verifikation und Validierung

Derzeit sind **MISRA** und **ISO 26262** die Software Standards für die Verifikation und Validierung von fahrzeugbasierender Software.

MISRA

Seit der Geburt des Motor Industry Software Reliability Association (MISRA) C Standards spielt die statische Analyse eine große Rolle bei der Entwicklung von Automobil-Anwendungen. ***„Guidelines for the Use of the C Language in Vehicle Based Software"*** ist ein Dokument, welches zuerst im Jahre 1998 veröffentlicht wurde, um die sichere Verwendung der Sprache C in der Automobilindustrie voranzutreiben. Es enthält Regeln, die eine Teilmenge der Sprache C definieren, welche heute weitgehend als Modell für einen guten Programmierstil akzeptiert werden. ***„MISRA C++: 2008 Guidelines for the use of the C++ Language in Critical Systems"*** wurde 2008 publiziert und definiert ähnliche Regeln für die Sprache C++.

ISO 26262

Die ISO 26262 ist momentan in der Entwicklung und trägt den Titel „Road vehicles -- Functional safety". Es ist ein Standard für die funktionale Sicherheit. Sie ist eine Adaption des Standards für funktionale Sicherheit IEC 61508 für elektrische/elektronische Fahrzeugsysteme. Kapitel 6 des ISO 26262 Standards behandelt die Empfehlungen für dynamisches Testen von Software und die Verifikation als Teil des Standards für die Softwareentwicklung.

Die empfohlenen Aktivitäten umfassen sowohl das Testen auf Komponenten- als auch auf Systemebene, sowie funktionelle Tests (bedarfsgerechte Tests und partitionierte Tests) und Tests der strukturellen Abdeckung.

Wie VectorCAST mit dem ISO 26262 Standard konform ist

Die embedded Testwerkzeuge für die ISO 26262 von VectorCAST stellen die Empfehlungen für das Testen von Software und die Verifikation im Teil 6 des Standards für Software-Entwicklung sicher. Dies wird erreicht indem es die Erstellung und das Management von Testfällen unterstützt um sicherzustellen, dass die Low-Level Softwareanforderungen getestet wurden.

VectorCAST wird auch für eine Vielzahl von Test-Aktivitäten zur Robustheit, wie der Bereichsprüfung und dem Out-of-Bounds Test eingesetzt.

Darüber hinaus unterstützt VectorCAST die Erfassung und Berichterstattung der strukturellen Code-Abdeckung für alle Automotive Safety Integrity Levels (ASIL) welche von ISO 26262 benötigt werden.

ASIL ist der Automobil-spezifische risikobasierte Ansatz zur Bestimmung von Risikoklassen bei Produkten. Risikoklassen werden von Level A bis D definiert, wobei ASIL D das höchste Risiko definiert.

Die VectorCAST/C++ Testwerkzeuge für das Testen von C und C++ Modulen und Integrationstests, kombiniert mit VectorCAST/Cover für die Test Verifikation auf Systemebene, bieten eine vollständig dynamische Testsuite für den Host, auf dem Simulator und Tests auf dem Zielsystem.

Bitte beachten Sie, dass die ISO 26262 derzeit in der letzten Phase vor der Anerkennung steht und sich die Empfehlungen sich vor der Genehmigung noch geringfügig ändern können.

Für dieses Dokument wird folgende Legende verwendet:

R	Empfohlene Aktivität
HR	Dringend empfohlene Aktivität

Die ISO 26262 schlägt einen Wasserfallansatz, mit einer klaren Abgrenzung zwischen den Modul-Test Aktivitäten und den System-Test-Aktivitäten für das Testen vor. Das Dokument behandelt daher diese zwei Ebenen des Testens getrennt voneinander.

VectorCAST beim Testen von Software-Modulen

Methoden für das Testen von Software-Modulen

Sektion 9 definiert die Ziele der Modul-Tests, sowie der Demonstration, dass die Software-Module die Spezifikationen erfüllen und keine unerwünschten Funktionalitäten beinhalten. Um dieses Ziel zu erreichen, empfiehlt der Standard, dass folgende Methoden für die Komponenten-Tests implementiert werden *(siehe ISO 26262 Tabelle 12 – Methoden für das Testen von Software-Einheiten.)*

Methoden	ASIL			
	A	B	C	D
1a. Anforderungsbasierter Test	HR	HR	HR	HR
1b. Test des Interfaces	HR	HR	HR	HR
1c. Test durch Fehlerinjektion	R	R	R	HR
1d. Test des Ressourcenverbrauchs	R	R	R	HR
1e. Back-to-Back Test zwischen Modell und Code (falls anwendbar)	R	R	HR	HR

ISO 26262 Tabelle 12 – Methoden für das Testen von Software-Einheiten

1a. Anforderungsbasierter Test

Dieser Test ist eine Methode, bei der Anforderungen an die Software, wie zum Beispiel die Funktionalität, Robustheit usw. zuerst durchgeführt werden. Dann erst wird der Code implementiert und Testfälle werden mit bestimmten Anforderungen verknüpft um zu zeigen, dass der Code diese erfüllt. Alle testbaren Anforderungen sollten getestet werden.

Der Anforderungstest ist auf allen Ebenen von ASIL dringendst empfohlen und ist der Grundstein der meisten Softwarestandards in kritischen Industrien, inklusive der der Avionikindustrie.

VectorCAST/C++ für Modul-Tests macht bedarfsgerechte Tests so effizient wie nur möglich. Eine Testumgebung besteht aus dem zu testenden Code und dem damit assoziierten Testcode der üblicherweise in Form eines Treibers oder Stubs vorliegt. Die Generierung der Testumgebung wird mit VectorCAST automatisiert.

Die Erstellung eines Anforderungstests wird durch den Einsatz von VectorCAST/C++ und der damit verbundenen Verwendung einer GUI die die Eingabe von Werten, Parametern, Rückgabewerten, globalen Variablen und sogar Stubs, sehr erleichtert. Stubs bestehen aus einem Stück Code und ersetzen Abhängigkeiten. Sie ermöglichen dadurch einen höheren Grad der Isolation. Die Erstellung von Testfällen kann auch durch das Importieren von Daten durch eine Datei mit z.B. durch Komma getrennten Werten (z.B einer CSV Datei) erfolgen.

Unter Verwendung des VectorCAST/Requirements-Gateways (ein Modul von VectorCAST/C++) kann jeder Testfall mit einer bestimmten Software-Anforderung verknüpft werden, welche dann über ein entsprechendes Werkzeug für das Anforderungsmanagement, wie beispielsweise DOORS, heruntergeladen werden kann. Sobald die Verbindung einmal hergestellt wurde, können diese Informationen zusammen mit den Statusinformationen der Tests auch zu einer Anforderungs-Datenbank geladen werden. Dies macht es noch einfacher zu überschauen, welche Anforderungen erfüllt wurden und welche nicht.

1b. Test des Interfaces

Die ISO 26262 empfiehlt dringendst den Test der Interfaces aller Module auf allen ASIL Ebenen. Dies kann auf verschiedene Wege erreicht werden, beispielsweise unter Verwendung von Grenzwerten bei Interfaces, unzulässigen Werten oder Mittelwerten. Diese Werte können auch auf verschiedene Art und Weisen miteinander kombiniert werden. Hierbei ist es das Ziel sicherzustellen, dass das Interface robust ist um das Auftreten von Fehlern zu vermeiden.

All diese Aktivitäten können mit VectorCAST/C++ einfach durchgeführt werden. Extremwerte können nach Funktionaler Größe oder Typ spezifiziert werden und unzulässige Werte können einfach angegeben werden. Der genaue Bereich der einzelnen Datentypen wird mit VectorCAST/C++ ordnungsgemäß in der Zielumgebung getestet, so dass die Grenzwerte präzise sind und für das automatische Durchführen von Grenzwert-Tests verwendet werden können. Dies kann zudem in einem kombinatorischen Modus durchgeführt werden, durch den effektive Testfälle aus einem einzelnen Datensatz vielfach durchgeführt werden können – völlig automatisch.

1c. Test durch Fehlerinjektion

Modultests können auch mit der bewussten Einführung von Fehlern, inklusive korrumpierter Werte bestimmter Variablen, durchgeführt werden. Dies dient dazu, die Sicherheitsmechanismen, die in der Applikation enthalten sind, zu testen. Diese Vorgehensweise wird speziell für ASIL D empfohlen. VectorCAST/C++ bietet alle Möglichkeiten für diese Art des Testens. Die Stub-Funktionalität, welche sich selbst auf Bibliotheksaufrufe erweitern lässt, kann auch verwendet werden, um absichtlich Fehler innerhalb einer Funktion zu injizieren. Wenn eine höhere Flexibilität erforderlich ist, kann der Testfall auch unter der Kontrolle des zugrunde liegenden Debuggers ausgeführt werden, so dass Variablenwerte zu verschiedenen Zeitpunkten an unterschiedlichen Stellen verändert werden können.

1d. Test des Ressourcenverbrauchs

Das Testen des Ressourcenverbrauchs auf Modulebene kann unter Verwendung des VectorCAST Testwerkzeugs für Komponententests für C und C++ teilweise automatisiert werden. Unter Verwendung des VectorCAST/RSP Moduls können Tests auf einem Zielgerät (oder einem Simulator) durchgeführt werden. Mit Hilfe von zusätzlichen Werkzeugen kann dann der Ressourcenverbrauch ermittelt werden.

1e. Back-to-Back Tests zwischen Modell und Code

Die Werkzeuge von VectorCAST werden bereits lange und erfolgreich verwendet, um Code zu testen, der von Modellen in Simulink® und Rhapsody® automatisch generiert wurde. Vector Software steht mit den Produzenten dieser Produkte in engem Kontakt um die Unit-Tests durch automatisch generierten Code so nahtlos wie möglich zu gestalten.

Methoden zur Ableitung von Testfällen für das Testen von Software Einheiten
Zusätzlich zu den verschiedenen Methoden für Unit-Tests, empfiehlt ISO 26262 auch vier verschiedene Strategien, um Testfälle während Software-Tests zu generieren. Diese sind wie folgt aufgeführt *(siehe ISO 26262 Tabelle 13 – Methoden zur Ableitung von Testfällen für das Testen von Software Einheiten).*

Methoden	ASIL			
	A	B	C	D
1a. Anforderungsanalyse	HR	HR	HR	HR
1b. Erzeugung und Analyse von Äquivalenzklassen	R	HR	HR	HR
1c. Analyse von Grenzwerten	R	HR	HR	HR
1d. Fehlereinschätzung	R	R	R	R

ISO 26262 Tabelle 13 – Methoden zur Ableitung von Testfällen für das Testen von Software Komponenten

Anforderungsbasiertes Testen auf Komponentenebene ist für die Erfüllung der ISO 26262 Richtlinie von zentraler Bedeutung und wird auch für alle ASIL Ebenen dringendst empfohlen.

Der Standard enthält zudem Bestimmungen für ein Testen der Code-Struktur selbst (Erstellung und Analyse von Äquivalenzklassen und Analyse von Grenzwerten) und wird empfohlen, wenn ein System auf den Ebenen ASIL B, ASIL C und ASIL D getestet wird.

1a. Anforderungsanalyse
Wie in den vorhergehenden Sektionen über das anforderungsbasierte Testen erwähnt, können Testfälle für Komponententests aus Low-Level Anforderungen bereitgestellt werden. In VectorCAST können solche Testfälle an spezifische Bedingungen geknüpft werden und ihr Status kann an ein eventuell vorhandenes Anforderungs-Managementsystem exportiert werden.

1b. Erzeugung und Analyse von Äquivalenzklassen
Diese Strategie zielt auf das angemessene Testen der Software ab, indem Partitionen auf der Eingangsdomäne bestimmt werden, die für die Ausführung der Software notwendig sind. Diese Testfälle zielen darauf ab, das Programm ausreichend zu testen. Sie können entweder auf Software-Spezifikationen, der internen Struktur des Programms oder auf beides basieren.

Diese Aktivitäten sind in VectorCAST/C++ voll automatisiert damit Anwender schnell Testfälle basierend auf Listen von Werten erstellen können. Diese Eingaben können in einem nicht-kombinatorischen, linearen Modus getätigt werden oder das Tool kann alle Eingabekombinationen verwenden, um Tests durchzuführen. Die Ausführung dieser komplexen Testfälle läuft automatisch ab, egal ob sie auf einem Host, in einem Simulator oder in der Zielumgebung durchgeführt werden.

VectorCAST hat zudem einen Partitions-Testfallgenerator, um automatisch zusätzliche Testfälle einer bereitgestellten Domäne zu erstellen.

1c. Analyse von Grenzwerten

Diese Tests zielen darauf ab, mögliche Softwarefehler bei Parametergrenzen oder Grenzwerten zu entfernen, mit denen die Tests höchstwahrscheinlich fehlschlagen. Nach ISO 26262 Tabelle 13 sollen zu testende Werte solche enthalten, die die Grenzen eines Interface erreichen oder überschreiten.

Dies kann mit VectorCAST/C++ sowohl auf Variablentypen als auch für den Einsatzbereich sehr einfach realisiert werden. Eine weitere Automatisierung wird dadurch erreicht, indem automatisch MIN-MID-MAX Testfälle erstellt werden, bei denen alle Werte auf das Minimum, den Medianwert und das Maximum gesetzt werden. Die minimalen und maximalen Werte werden ermittelt, indem die Reichweite aller im Programm verfügbaren Typen auf dem Zielsystem oder einem Simulator getestet werden. Daraus folgt: Durch den Einsatz des Tools auf dem Zielgerät oder dem Simulator kann garantiert werden, dass die Größe der Grenzwerte durch die automatisch erstellten MIN-MID-MAX Tests für das System Gültigkeit hat, egal ob es sich um ein 8, 16 oder 32-Bit System handelt.

Diese zwei Werkzeuge können auch annähernde, unzulässige und sogar spezielle Werte wie Not-A-Number (NaN), positive und negative Unendlichkeit bei Gleitpunktvariablen überprüfen.

1d. Fehlereinschätzung

Nach ISO 26262 Tabelle 13 können diese Tests auf Daten basieren, die durch einen Prozess erlernt und mittels professioneller Beurteilung gesammelt wurden. Diese werden für ASIL Stufe A, B und C empfohlen und werden dringendst anzuwenden, wenn das System auf Einhaltung der Anforderungen für ASIL Stufe D getestet werden soll. Sie zielen darauf ab die Komponenten der Software so fehlerfrei wie möglich zu erstellen.

Mit VectorCAST ist das Erstellen eines Testfalles eine einfache Angelegenheit und man muss hierzu keine unübersichtlichen Skripte verwenden. Signale können Exceptions ausgeben und Zeiger können freiwillig un-initialisiert belassen werden, um zu sehen ob der Code dagegen geschützt ist. „Was wäre wenn“ Szenarien können einfach erstellt werden, während manuelle Tests oder Skript-basierte Werkzeuge erhebliche Mengen an zu entwickelnden Testcode erfordern würden.

Alle Testfälle werden außerhalb der Testumgebung gehalten, bis sie benötigt werden. Das bedeutet, dass Testfälle erstellt und gelöscht werden können, ohne den Code erneut zu kompilieren. Das maximiert die Produktivität im Vergleich zu Werkzeugen anderer Hersteller.

Strukturelle Metriken zur Erfassung auf Software Unit Level

Die ISO 26262 empfiehlt den Einsatz der Codeabdeckung als Metrik, um zu messen, ob auf einem bestimmten Modul eine ausreichende Überprüfung stattgefunden hat. Da es fast unmöglich ist eine 100% Codeabdeckung zu erreichen, wenn das System als Ganzes getestet wird (durch Erfassung von Statements), empfiehlt die ISO 26262, dass zumindest jede Codezeile während dem Modul-Test ordnungsgemäß überprüft wird.

Das Ziel ist eine 100% Abdeckung basierend auf den gewählten Erfassungskriterien zu erreichen. Gemäß ISO 26262 Tabelle 14, Anm. 4, wird eine rationale Zahl für den Grad der erreichten Abdeckung verwendet (z.B. für entgegengenommenen toten Code oder Codesegmente, die von verschiedenen Software-Konfigurationen abhängig sind), ansonsten kann nicht abgedeckter Code unter Verwendung komplementärer Methoden verifiziert werden (Bsp. Kontrollen).

Auf Modulbene kann aus drei verschiedenen Kriterien für die Codeabdeckung gewählt werden (*siehe ISO 26262 Tabelle 14 -strukturelle Metriken zur Erfassung auf Software Einheitsebene*)

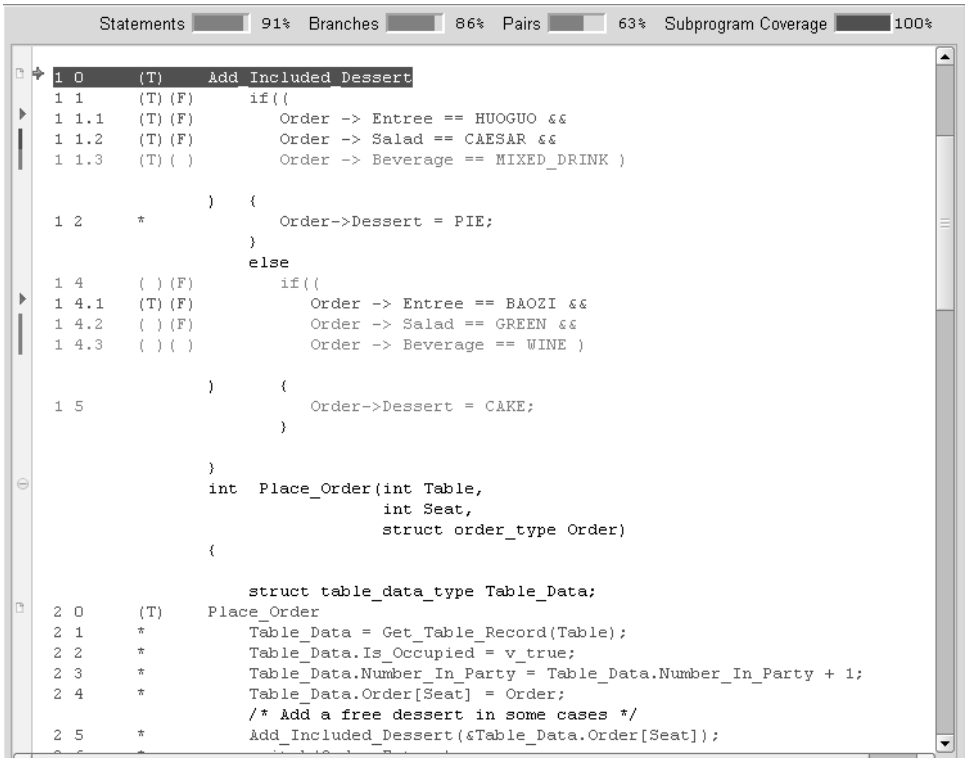
Methoden	**ASIL**			
	A	**B**	**C**	**D**
1a. Statement Coverage	HR	HR	HR	HR
1b. Branch Coverage	R	HR	HR	HR
1c. MC/DC (Modifizierte Bedingung/ Entscheidungserfassung)	R	R	R	HR

ISO 26262 Tabelle 14 – Strukturelle Metriken zur Erfassung auf Software Einheitsebene

Es sei anzumerken, dass diese Ebenen der Erfassung deutlich schwieriger zu erreichen sind. Bei Verwendung eines Tools wie VectorCAST (welches den Anwendern das Testen von MC/DC, Branch und Statements isoliert ermöglicht) sollten die folgenden Kombinationen von Kriterien verwendet werden, um einen Level der strukturellen Erfassung zu erreichen, der mit den in ISO 26262 definierten Normen übereinstimmt.

- *Statement Erfassung nur bei ASIL A*
- *Statement und Branch Erfassung bei ASIL B und C*
- *Statement, Branch, **and** MC/DC for ASIL D*

VectorCAST/C++ (sowie VectorCAST/Cover) hat eine einfach zu bedienende Anzeige für die Codeerfassung. Die Anzeige zeigt durch Symbole und Farbcodes ob der Code (a) vollständig erfasst (in grün), (b) teilweise erfasst (in orange), oder (c) gar nicht erfasst (rote Farbe) wurde. Indem der Mauszeiger auf einer der Codezeilen belassen wird, kann der Benutzer erkennen, welche Testfälle von diesen Zeilen abgedeckt wurden.



Im Falle von Zielumgebungen mit ausreichend Speicher und liberalen Zeitbedingungen kann die Codeerfassung sogar animiert werden – wörtlich „abspielen“ wie die Codeerfassung während der Ausführung erreicht wurde. Bei eingeschränkteren Umgebungen kann die Codeerfassung auch in Modi ausgeführt werden, welche weniger Speicherplatz und/oder weniger Einfluss auf Timing-Probleme haben. Die Instrumentierung der VectorCAST Codeerfassung und Datenerfassung hat verschiedene Optionen, die es dem Benutzer erlauben, auf maximale Ressourceneffizienz für spezielle Anwendungen anzupassen.

1a. Statement Erfassung

Die Statement Erfassung versucht individuelle Codezeilen in einer gegebenen Einheit abzudecken. Beispielsweise würde bei folgender Codezeile ein einzelner Testfall abgedeckt werden müssen.

```
if(i < 10 && j == 0 || k > 12)
```

Es sei darauf hingewiesen, dass obwohl die Bewertung eines einzelnen Testfalles bei dieser Zeile auf „False“ gesetzt ist, zusätzliche Zeilen innerhalb des 'IF' Statements enthalten sein können. Wenn dieses 'IF' Statement ebenfalls ein 'ELSE' Statement aufweist, würde ein Testfall mit der Bewertung auf „True“ nicht den Inhalt des 'ELSE' Statements abdecken. Allerdings decken sowohl der „True“ als auch der „False“ Test das 'IF' Statement ab.

VectorCAST/C++ (sowie VectorCAST/Cover) unterstützt Statement Erfassung, sowohl als einzelne Lösung oder in Kombination mit Branch und/oder MC/DC Erfassung. Zudem können die Möglichkeiten der automatischen Testfallerstellung in VectorCAST/C++ den Entwicklern dabei helfen, Testfälle zu ermitteln, um die Statement Erfassung zu maximieren. Beispielsweise liefern automatisch generierte Testfälle basierend auf Grundpfaden (die eindeutigen Pfade innerhalb des Codes) oft einen höheren Grad der Statement Erfassung.

1b. Branch Erfassung

Die Branch Erfassung konzentriert sich auf den Status der Entscheidungspunkte, die entweder wahr oder falsch sein können. Die folgende Codezeile würde beispielsweise mit zwei Testfällen abgedeckt werden müssen – einer bei dem das 'IF' Statement einen wahren Wert zurück gibt und einen, in dem es als falsch betrachtet wird.

```
if(i < 10 && j == 0 || k > 12)
```

VectorCAST/C++ und auch VectorCAST/Cover liefern eine vollständige Unterstützung von Branch Erfassung, entweder als Standalone Anwendung oder in Kombination mit anderen Kriterien der Codeerfassung. Um den Anforderungen von ISO 26262 gerecht zu werden, kann VectorCAST sowohl die Statement als auch die Branch Erfassung während eines einzelnen Testlaufs durchführen.

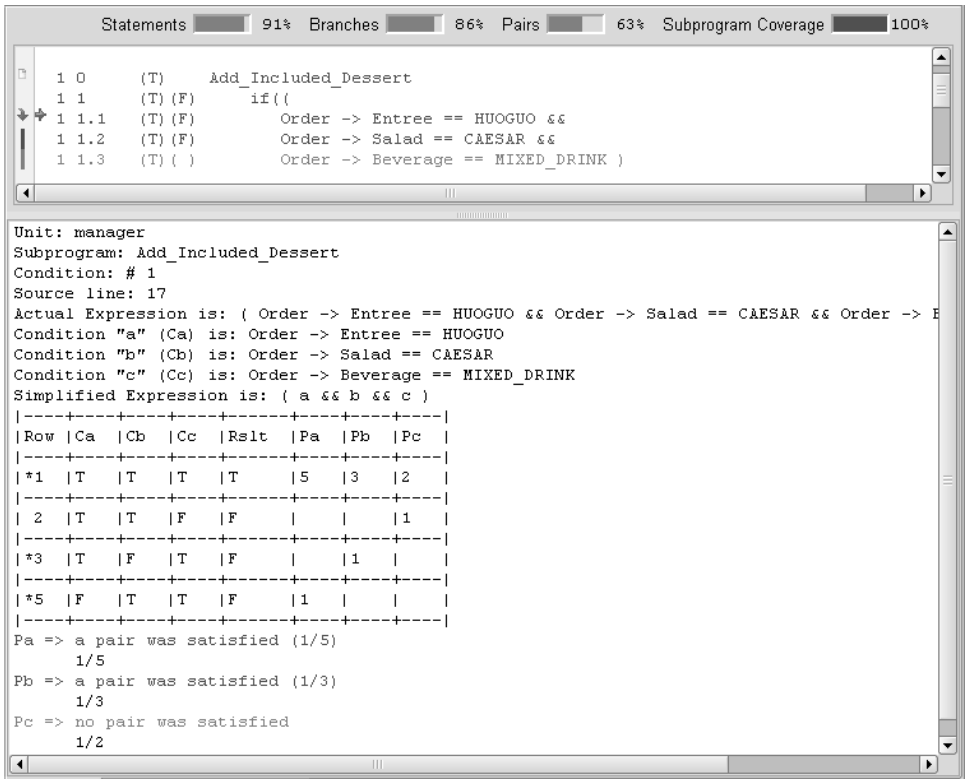
1c. MC/DC (Modifizierte Bedingung/Entscheidungserfassung)

MC/DC erfordert die größte Anzahl an Tests, um den Anforderungen zur Erfassung gerecht zu werden. Dieses Niveau der Erfassung demonstriert, dass alle Unterbedingungen, die Teil einer bedingten Anweisung sind, das Ergebnis einer bedingten Anweisung beeinflussen können. Beispielsweise im Falle des folgenden Statements:

```
if(i < 10 && j == 0 || k > 12)
```

Es soll gezeigt werden, dass sich das Endergebnis durch eine Änderung des Wertes von 'I' ohne weitere Änderungen an anderen Variablen, ändern wird.

Diese Aufgabe kann auch für den erfahrenen Ingenieur sehr anstrengend sein. VectorCAST liefert allerdings eine sehr effiziente Möglichkeit, diese Art des Testens unter Verwendung des VectorCAST/MCDC Moduls durchzuführen. Es wird automatisch eine Wahrheitstabelle erstellt, welche deutlich aufzeigt, welche Testfall-Paarungen für die MC/DC Erfassung benötigt werden und kennzeichnet dann, welche Testfälle und Testfall-Paarungen zur Verfügung gestellt wurden.



Ausführung auf dem Zielgerät oder einem Simulator

Die ISO 26262 zeigt eine starke Präferenz für die Ausführung von Modultests in einer Umgebung, die der Umgebung so nahe wie möglich kommt in der das Programm in dem Gerät betrieben wird. Wie bereits in Sektion 9.4.5 erwähnt, *sollte die Testumgebung für das Testen von Software Modulen so weit wie möglich der Zielumgebung entsprechen. Wenn das Testen der Softwaremodule nicht in der Zielumgebung durchgeführt wird, müssen die Unterschiede im Quell- und Objektcode und die Unterschiede zwischen der Testumgebung und der Zielumgebung analysiert werden, um zusätzliche Tests auf der Zielumgebung während späteren Testphasen zu spezifizieren.*

VectorCAST/C++ in Verbindung mit dem VectorCAST/RSP (Runtime Support Package) ist das ideale Werkzeug um diesem Teil der ISO 26262 zu entsprechen. Durch den Einsatz des RSP Moduls kann VectorCAST automatisch Testfälle auf dem Zielgerät durchführen oder (im Falle ressourcenarmer Umgebungen) simulieren. VectorCAST/RSP wurde speziell für den Einsatz mit einem speziellen Satz von Compiler, Bord, Debugger und RTOS (falls vorhanden) Kombinationen erstellt, welche es erlaubt, auf dem eigentlichen Ziel einen Testfall mit wenig oder gar keinen Benutzereingaben laufen zu lassen. Die erneute Ausführung von Testfällen (Regressionstests) von der Kommandozeile aus ist ebenfalls 100% automatisch durchführbar.

VectorCAST während der Integration und dem Softwaretest

Sektion 10 definiert bestimmte Integrationsebenen, die gegen den Entwurf der Softwarearchitektur getestet wurden. Diese Integrationsebenen basieren auf der hierarchischen Architektur der Software – von ganz unten (Testen individueller Module) bis ganz oben (Testen der Software als Ganzes).

Obwohl die ISO 26262 hier nicht unterscheidet ist das Testen der Module (Einheiten, die miteinander kombiniert werden, um den funktionalen Prozess zu repräsentieren, anstatt die gesamte Software) ein großer Vorteil. Der bei Vector Software www.vectorcast.com verwendete Begriff ist M*odultest oder Integrationstest*. Ein Test der auf der gesamten Applikation ausgeführt wird, nennt man häufig einen *Systemtest*.

In der ISO 26262 besteht der Prozess aus der schrittweisen Integration von verschiedenen Modulen, basierend auf der jeweiligen Hierarchie der Software, *bis die embedded Software vollständig* auf Systemebene *integriert* ist (Sektion 10.4.1). Daher wird empfohlen, dass sowohl Modultests als auch Systemtests durchgeführt werden.

Methoden für die Softwareintegration und das Testen abgeleiteter Testfälle

Genau wie während der Modultestphase, wird eine Vielzahl von Methoden für das Testen der Softwareintegration empfohlen. Diese sind unten aufgeführt (*siehe ISO 26262 Tabelle 15 – Methoden für das Testen von Softwareintegration*)

Methoden	ASIL			
	A	B	C	D
1a. Bedarfsgerechter Test	HR	HR	HR	HR
1b. Testen des Interface	HR	HR	HR	HR
1c. Test durch Fehlerinjektion	R	R	HR	HR
1d.Test des Ressourcenverbrauchs	R	R	R	HR
1e. Back-to-Back Test zwischen Modell und Code (falls zutreffen)	R	R	HR	HR

ISO 26262 Tabelle 15 – Methoden für das Testen von Softwareintegration

Die Methoden für abgeleitete Testfälle für Softwareintegrationstests sind unten aufgelistet (*siehe ISO 26262 Tabelle 16 – Methoden für abgeleitete Testfälle für das Testen der Softwareintegration*)

Methoden	ASIL			
	A	B	C	D
1a. Anforderungsanalyse	HR	HR	HR	HR
1b. Erzeugung und Analyse von Äquivalenzklassen	R	HR	HR	HR
1c. Analyse von Grenzwerten	R	HR	HR	HR
1d. Fehlereinschätzung	R	R	R	R

ISO 26262 Tabelle 16 – Methoden für abgeleitete Testfälle beim Testen von Softwareintegration

Diese Methoden sind exakt dieselben wie in der Sektion über das Testen von Modulen definiert. In dieser Sektion konzentrieren wir uns darauf, die Features von VectorCAST hervorzuheben, welche sich speziell an Integrationstests richten.

Wie man eine Testumgebung für Integrationstests erstellt

VectorCAST/C++ kann Integrationstests während des Modul/Integrationstests ausführen. Die Verwendung ist dieselbe wie bei der Testphase von Modultests – der einzige Unterschied ist, dass anstatt einer einzigen Datei nun mehrere Dateien für den Testfall verwendet werden.

VectorCAST/C++ bietet echte Integrationstests – Die Dateien werden zu einem größeren integrierten Modul zusammengefasst. Daraus folgt, dass Tests auf einer Funktion des ersten Moduls ausgeführt werden und dieses Modul eine andere Funktion des zweiten Moduls aufruft, beide Funktionen getestet werden und für beide eine Codeabdeckung erstellt wird. Wie im Falle von Modultests, muss kein Skript geschrieben werden um eine Testumgebung für Integrationstests zu erstellen – alles wird automatisch erstellt.

Wiederverwendung von Testfällen von Modultests

Die erweiterten Fähigkeiten von VectorCAST zur Regression können sich auch während des Integrationstests als nützlich erweisen. Grundsätzlich ist es möglich, Teile oder alle Testfälle wiederzuverwenden, die während des Modultests erstellt wurden um diese nahtlos bei den Integrationstests wieder anzuwenden. Der Code kann sich eventuell geändert haben, jedoch werden die Testfälle ohne Probleme laufen (solange das Interface der Funktion stabil bleibt, ansonsten wird der Testfall verworfen oder als nicht mehr relevant gekennzeichnet). Diese Fähigkeit Testfälle zu importieren und exportieren bietet eine enorme Zeitersparnis.

Testen von Softwaremodulen (Bibliotheken) von Drittanbietern

Zusätzlich kann VectorCAST/C++ verwendet werden, um Bibliotheken zu testen und dies sogar ohne Zugriff auf den Code zu haben. Dies kann sich ebenfalls als nützlich erweisen, wenn man Softwarekomponenten integrieren möchte, die von Drittanbietern kommen, wie zum Beispiel Lieferanten und Auftragsnehmer. Das Erstellen eines Testfalles um eine Bibliothek ist so einfach wie das Erstellen eines Tests bei bestehendem Code und die Testaktivitäten können auf ähnliche Weise ausgeführt werden.

Systemtests

Sobald die Integration die Systemebene erreicht, wird der Eingang von Testfällen durch andere Mittel, wie beispielsweise der Signalerzeugung, das Drücken von Knöpfen auf der Konsole oder eventuell mit Hilfe eines Simulators vorgegeben. Während des Systemtests kann VectorCAST/Cover für die Ermittlung der Codeabdeckung verwendet werden (siehe die beiden unteren Sektionen).

Strukturelle Erfassungsmetriken auf Ebene der Softwarearchitektur

Methoden	ASIL			
	A	B	C	D
1a. Funktionserfassung	R	R	HR	HR
1b. Erfassung von Aufrufen	R	R	HR	HR

ISO 26262 Tabelle 17 – Strukturelle Erfassungsmetriken auf Ebene der Softwarearchitektur

Die Funktionsabdeckung wird durch jede Funktion definiert, die mindestens einmal während des Testlaufes aufgerufen wird und mindestens eine Ausführung jedes Funktionsaufrufs benötigt. Sektion 10.4.6 legt weiterhin fest, dass die nicht spezifizierten Softwarekomponenten identifiziert und entfernt oder deaktiviert werden müssen.

VectorCAST/Cover (wie auch VectorCAST/C++) hält die erreichte Funktionsabdeckung fest. Zudem kann eine Erfassung von Aufrufen indirekt erreicht werden, indem eine partielle Statement-Erfassung durchgeführt wird, welche alle Aufrufe jeglicher Funktionen in der Integration oder der Testumgebung des Systems stimuliert.

Ausführung von Integrationstests

Wie zuvor beschrieben, empfiehlt die ISO 26262 für Modulstests Die Ausführung von Testfällen in einer Umgebung hin, welche der wirklichen Umgebung, in der das Programm zum Laufen gebracht werden soll, so nahe wie möglich kommt. Dies kann sowohl unter Verwendung von VectorCAST/C++ als auch VectorCAST/Cover realisiert werden.

Eine häufig gestellte Frage zu VectorCAST/Cover ist, wie das Werkzeug die Daten aus der Codeerfassung von verschiedenen Zielen exportiert. Das Werkzeug kann die Daten der Erfassung durch drei verschiedene Methoden exportieren: (a) in eine Datei speichern (nützlich im Falle, dass ein Dateisystem vorhanden ist oder eine simulierte Datei I/O aufgebaut werden kann), (b) durch einen Port senden (wie einem seriellen Port) und (c) die Daten in einem Zwischenspeicher halten, welcher dann mit Hilfe des Debuggers oder eines anderen Werkzeugs gespeichert werden kann.

Berichterstattung

Die ISO 26262 spezifiziert die Erstellung einer Vielzahl von Dokumenten, wie die Spezifikation der Softwareverifikation und einem Verifikationsbericht der Software.

Die VectorCAST Produkte erstellen eine Vielzahl von Testergebnisse, basierend auf Modultests, Integrationstests oder Systemtests. Diese Berichte können entweder im Text oder HTML Format erstellt werden. Sie können außerhalb von VectorCAST gespeichert werden und werden verwendet, um einer Vielzahl von Standards gerecht zu werden, so wie: IEC 61508, CENELEC, DO-178B usw.

Zertifizierung

Sowohl VectorCAST/C++ als auch VectorCAST/Cover können für die Einhaltung von Aktivitäten zertifiziert werden, die den Anforderungen von ISO 26262 gerecht werden. Bitte kontaktieren Sie Vector Software für weitere Informationen.

Weitere Werkzeuge

Abschließend kann VectorCAST/C++, VectorCAST/Cover und das VectorCAST/Requirements Gateway das Testen und die Codeerfassung automatisieren, um den ISO 26262 Anforderungen effizienter gerecht zu werden. All diese Werkzeuge können ihre individuellen Berichte als HTML oder Text exportieren, die schon häufig erfolgreich eingesetzt wurden, um einer Vielzahl anspruchsvoller Industriestandards gerecht zu werden.

VectorCAST/RSP ist das Modul, welches die Ausführung eines Tests auf einem Simulator oder einem Zielgerät ermöglicht. Dieser Prozess ist vollständig automatisiert, so dass Testfälle individuell oder als Gruppe durch einen einfachen Mausklick oder der Befehlszeile ausgeführt werden können. Die Ausführung an sich benötigt keine Eingaben durch den Benutzer.

VectorCAST/Manage bietet die Möglichkeit vollständig automatischer Regressionstests für alle durch VectorCAST erstellten Tests.

Erfolge auf sicherheitskritischen Märkten

Die VectorCAST Werkzeuge werden seit mehr als 20 Jahren erfolgreich von Unternehmen eingesetzt, welche sicherheitskritische- und unternehmens-kritische Anwendungen entwickeln. Dazu gehören zum Beispiel: Militär, Luftfahrtelektronik, Luft- und Raumfahrt, medizinische Geräte, Bahn, Automobile und Industriesteuerungen. Hunderte von Unternehmen verwenden VectorCAST um den Anforderungen beim Testen eingebetteter Software zu erfüllen, welche bald nach ISO 26262 auf dem Automobilmarkt Verwendung finden.

Über Vector Software

Vector Software Inc. Ist der führende unabhängige Anbieter von automatisierten Testwerkzeugen für Softwaretests von sicherheitskritischen Anwendungen: Vector Software's VectorCAST Produkte automatisieren und verwalten die komplexen Aufgaben in Verbindung mit Modulen, Integration und dem Testen auf Systemebene. Die VectorCAST Produkte unterstützen die C, C++ und Ada Programmiersprachen.

Vector Software, Inc.
1351 South County Trail, Suite 310
East Greenwich, RI 02818
USA
P: 401.398.7185
F: 401.398.7186
E: info@vectorcast.com
W: vectorcast.com